UNIVERSITÀ DI PARMA
Dipartimento di Ingegneria e Architettura

Secret Key (symmetric) Cryptography

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Course of Cybersecurity, 2022/2023

http://netsec.unipr.it/veltri

Cryptography

- Study of mathematical techniques related to information and communication security in the presence of third party adversaries
- The most widely used tool used by different security services
  - **not the only one**
- Can be used for:
  - **Confidentiality**
  - **Data integrity**
  - **Authentication**
  - **Non-repudiation**

Different cryptographic algorithms

- Symmetric cryptography (Secret key cryptography)
  - **the two communication parties share a common secret (key)**

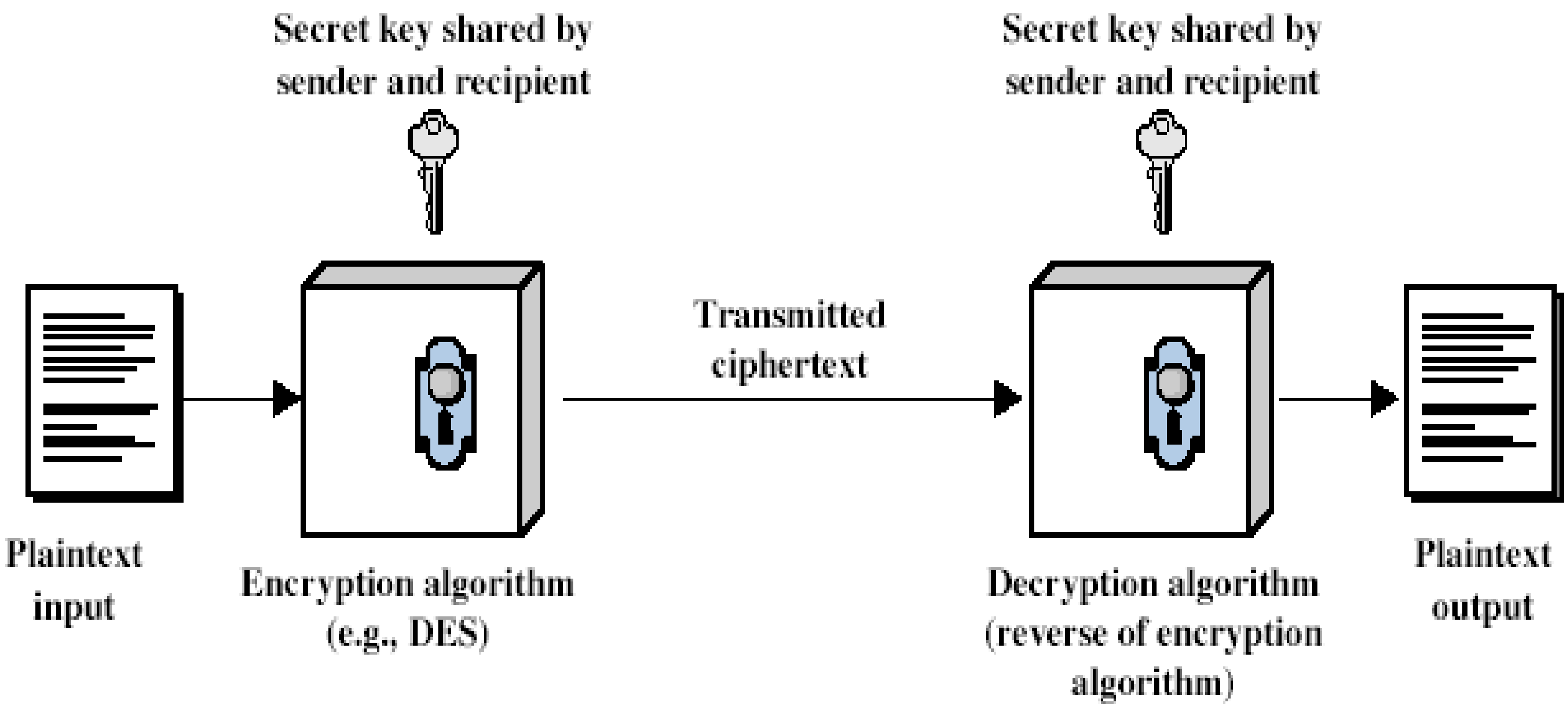
- Asymmetric cryptography (Public key cryptography, or private/public key cryptography)
  - **two different keys are used for two opposite functions (e.g. encryption and decryption)**
  - **one key can be publicly available (public key); the other is maintained secret for the owner (private key)**

- Hash algorithm (message digest/one way transformation)
  - **one-way transformation that maps a variable length message to a fixed length bit string**
  - **a variant is a MAC function**
    - includes a key

Symmetric Cryptography

- Or conventional / secret-key / single-key
  - **sender and recipient share a common key**
- All classical encryption algorithms are secret-key
  - **was the only type prior to invention of public-key in 1970's**
- Generally used for protecting (through encryption) some data stored in a repository or sent to a remote entity
- Designed to take a reasonable-length key (e.g. 128 bits) and generating a one-to-one mapping from cleartext to ciphertext that "looks like completely random", to someone doesn't know the key

Symmetric Cipher Model



Symmetric Cipher Model (cont.)

- Plaintext - the original message (m)
- Ciphertext - the encoded message (c)
- Key - info used known only to sender/receiver (k)
- Cipher - algorithm for transforming plaintext to ciphertext

- Two functions:
  - **Encipher (Encryption) - converting plaintext to ciphertext**
    - $c = E(k,m) = E_k(m)$
      - can be either a deterministic or randomized function
  - **Decipher (Decryption) - recovering ciphertext from plaintext**
    - $m = D(k,c) = D_k(c) = D_k(E_k(m))$
      - deterministic

- Common symmetric algorithms:
  - **DES, 3DES, RC4, IDEA, AES**

Symmetric Cipher Model (cont.)

- In theory, the security of a cipher might rest in the secrecy of its restricted algorithm
    - c = E(m), m = D(c)

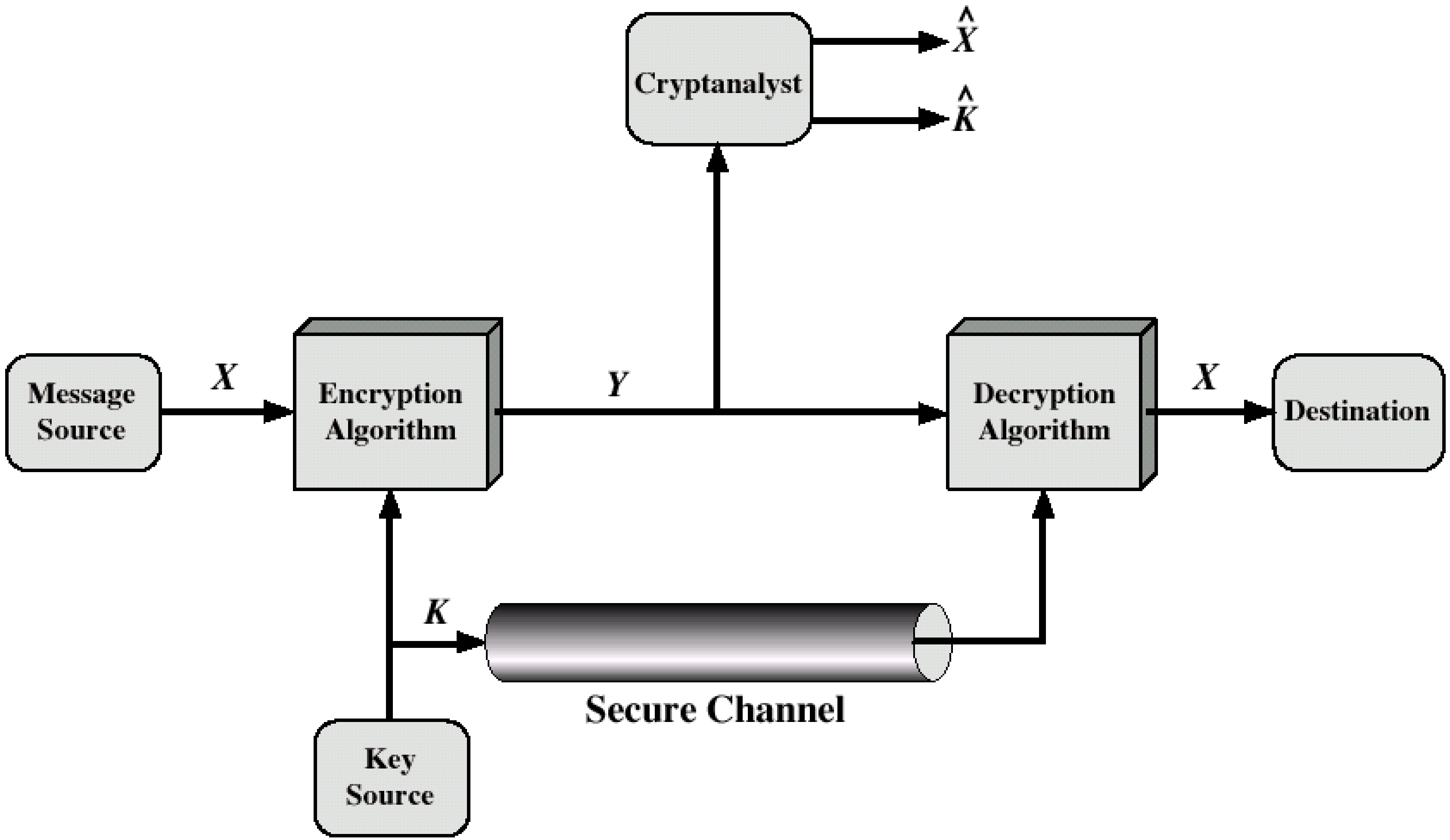
  however:
    - whenever a user leaves a group, the algorithm must change
    - could be scrutinized by people smarter than you

- In practice, the encryption algorithm is usually not secret
  - **keys are used and the security relies on the secrecy of keys**
    - selected from a large set (a keyspace), e.g., a 256-bit number $\rightarrow 2^{256} \approx 10^{77}$ values!
    - $c = E(k,m) = E_k(m)$, $m = D(k,c) = D_k(c)$
    - change of authorized participants requires only a change in key
    - the robustness of the algorithm is usually proportional to the key length
      - e.g. 40 bit (weak), 128 bit (strong)
  - **Kerckhoffs' principle: Security should be based on secrecy of the key, not the details of the algorithm**
    - Jean Guillaume Hubert Victor Francois Alexandre Auguste Kerckhoffs von Nieuwenhof, "La Criptographie Militaire", 1883

Symmetric Cipher Model (cont.)

- The two parties must know the algorithm to be used and must share a secret key
  - **requires an initial phase where the two parties exchange in secure manner the shared secret key**
    - implies a secure channel (or method) to distribute the key

Symmetric Cipher Model (cont.)



Threat model

- This specifies what “power” the attacker is assumed to have, without placing any restrictions on the adversary’s strategy
- Plausible options for the threat model are:
  - **Ciphertext-only attack**
  - **Known-plaintext attack**
  - **Chosen-plaintext attack**
  - **Chosen-ciphertext attack**

Ciphertext only - Attack

- The bad guy has seen (and presumably stored) some ciphertext that can be analyzed
  - **he/she should be able to recognize when he/she has succeeded (often called *recognizable plaintext* attack)**
    - for example in case of normal text or known document formats
  - **it is necessary to have enough ciphertext**

- It is the hardest attack to carry on
  - **the opponent has the least amount of information to work with**

Known plaintext - Attack

- The bad guy knows a <plaintext, ciphertext> pair
- From that pairs, the attacker can try to figure out the mapping of some fraction of the text
- How it is possible to obtain the plaintext?
  - **the secret data does not remain secret forever (e.g. the name of an attacked city)**
  - **or the opponent may have knowledge of what is in the message**
    - certain key words (e.g. in the header of the message)
      - expected patterns (e.g. PostScript, etc.)
      - probable-word attack
        - » between *Ciphertext only* attack and *Known plaintext attack*
- Some cryptographic schemes might be good enough to be secure against *ciphertext only* attacks but not against to *known plaintext* attacks
  - **in these cases, it is important to minimize the possibility for a bad guy to obtain <m,c> pairs**

Chosen plaintext (or ciphertext) - Attack

- The opponent can choose any plaintext and get the corresponding ciphertext from the system (or the contrary)
  - **e.g. there is a transmission service that encrypts and transmits messages; the bad guy can ask the transmission service to transmit any plaintext he/she wants**

- Some cryptographic schemes might be good enough to be secure against *ciphertext only* attacks and *known plaintext* attacks but not against to *chosen plaintext* attacks

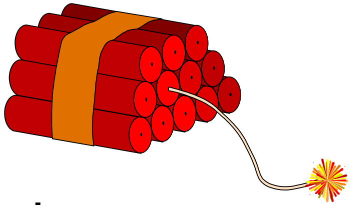
Cryptography Attacks

- There are some general approaches to attacking a conventional encryption scheme:
  - **Cryptographic analysis (cryptoanalysis)**
    - based on the type of the cryptographic algorithm, tries to exploit some characteristic of the algorithm and/or properties of some previous plaintext/ciphertext pairs to deduce a plaintext and/or key
      - does not require to obtain the encryption key for deduce the plaintext
  - **Brute-force (search) attack**
    - tries every possible encryption/decryption (e.g. by trying all possible keys)
      - it may require the visit of all key space
      - The average number of required attempts is the half of the number of possible keys
    - it requires to be able to recognize when the correct plaintext/ciphertext has been obtained
  - **Side-channel attacks**
    - use information from the physical implementation of a cryptosystem

Cryptoanalysis

- Based on the type of the cryptographic algorithm, tries to exploit some characteristic of the algorithm and/or properties of some previous plaintext/ciphertext pairs
  - **to deduce a plaintext and/or key**
  - **does not require to obtain the encryption key for deduce the plaintext**

Brute force attack



- Method of defeating a cryptographic scheme by trying a large number of possibilities
  - **for symmetric-key ciphers it typically means a brute-force search of the key space**
    - testing all possible keys in order to recover the plaintext used to produce a particular ciphertext
- In most schemes, the theoretical possibility of a brute force attack is recognized, but it is set up in such a way that it would be computationally infeasible to carry out
- The attacker has to determine when he succeeded
  - **By obfuscating the data to be encoded, brute force attacks are made less effective as it is more difficult to determine when one has succeeded in breaking the code**

Brute Force Search - Example

Key Size (bits)	Number of Alternative Keys	Time Required at 1 Decryption/μs	Time Required at 10^6 Decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu s$ = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu s$ = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu s = 6.4 \times 10^{12}$ years	6.4×10^6 years

Side channel attack

- Any attack based on information gained from the physical implementation of a cryptosystem, rather than theoretical weaknesses in the algorithms
  - **e.g. timing information, power consumption**
- General classes of side channel attack include:
  - **Timing attack — attacks based on measuring how much time various computations take to perform**
  - **Power monitoring attack — attacks which make use of varying power consumption by the hardware during computation**
  - **TEMPEST (aka Van Eck or radiation monitoring) attack — attacks based on leaked electromagnetic radiation which can directly provide plaintexts and other information**

Side channel attack (cont.)

- In all cases, that physical effects caused by the operation of a cryptosystem can provide useful extra information about secrets in the system
  - **about the cryptographic key, partial state information, full or partial plaintexts and so forth**
- Side-channel attacks require considerable technical knowledge of the internal operation of the system on which the cryptography is implemented

Cryptographic break

- A cryptographic "break" is anything faster than an exhaustive search (brute force attack)
  - **example, an attack against a 128-bit-key cipher requiring “only” 2^{120} operations (compared to 2^{128} possible keys) would be considered a break even though it would be, at present, quite infeasible**

- The loss of a key (also without cryptoanalysis or brute-force attack) is called a “compromise”

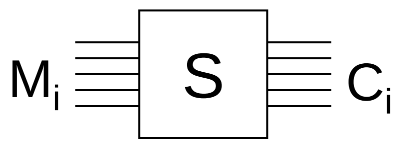
Computational and Unconditional Security

- Unconditional security
  - **an encryption scheme is unconditionally secure if no matter how much computer power is available, the cipher cannot be broken**
    - the ciphertext provides insufficient information to determine the corresponding plaintext
    - e.g. OTP (One Time Pad) chiper
- Computational security
  - **however cryptographic algorithms are often not impossible to attack**
    - e.g. brute force attack
  - **an encryption scheme is computationally secure if given limited computing resources the cipher cannot be broken**
    - e.g. the time required to break the cipher exceeds the useful lifetime of the information
    - depends on the attack complexity and cost
      - processing complexity: a large number of operations required (long time)
      - data complexity: a large number of expected inputs (e.g., ciphertext)
      - storage complexity: a large amount of storage units required
    - requires the estimation of computation power of the opponent

Classical encryption techniques

Substitution Ciphers

- Substitution is a classical encryption technique
- Letters of plaintext are replaced by other letters or by numbers or symbols
- If plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns



Substitution Table

M	***C***
0000	1101
0001	1001
0010	0111
0011	1000
⋮	⋮
1111	0011

- Simple examples of classical substitution ciphers
  - **monoalphabetic substitution with shift (e.g. Caesar cipher)**
  - **monoalphabetic substitution (monoalphabetic cipher)**
  - **polialphabetic substitution (polialphabetic cipher)**

Caesar Cipher

- Earliest known substitution cipher
  - **by Julius Caesar**
  - **first attested use in military affairs**
  - **it is a monoalphabetic substitution with shift**
- Replaces each letter by 3rd letter on
- Example:

```
meet me after the toga party
PHHW PH DIWHU WKH WRJD SDUWB
```

Caesar Cipher (cont.)

- Can define transformation (substitution) as:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

- Mathematically give each letter a number

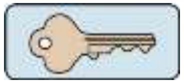
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

- Then have Caesar cipher as:

$C = E(P) = (P + k) \bmod 26$, with k=3

$P = D(C) = (C - k) \bmod 26$, with k=3

- If *k* is generic (and secret), we have a *Shift cipher*
  - **k is the key, with $K \in \{0,1,\ldots,25\}$**



Cryptanalysis of a Shift Cipher

- Only have 26 possible ciphers
  - **A maps to A,B,..Z**
  - **If the mapping of one letter is discovered, the entire transformation is found**
- Given ciphertext, could just try all shifts of letters
  - **brute force search**
  - **e.g. break ciphertext "GCUA VQ DTGCM"**
- Do need to recognize when have plaintext

KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

Monoalphabetic Substitution Ciphers

- Rather than just shifting the alphabet
- Could shuffle (permute) the letters arbitrarily
- Each plaintext letter maps to a different random ciphertext letter
- Example:
  - **Substitution table:**

`abcdefghijklmnopqrstuvwxyz`
`DKVQFIBJWPESCXHTMYAUOLRGZN`

  - **`plaintext: ifwewishtoreplaceletters`**
  - **`ciphertext: WIRFRWAJUHYFTSDVFSFUUFYA`**

- Note: the secret substitution can be seen as the secret key
- Now have a total of $26! \cong 4 \times 10^{26}$ keys
  - **with so many keys, might think is secure**
    - but would be wrong!
    - cryptoanalysis based on text frequency and correlation

Cryptoanalysis of Monoalphabetic Cipher

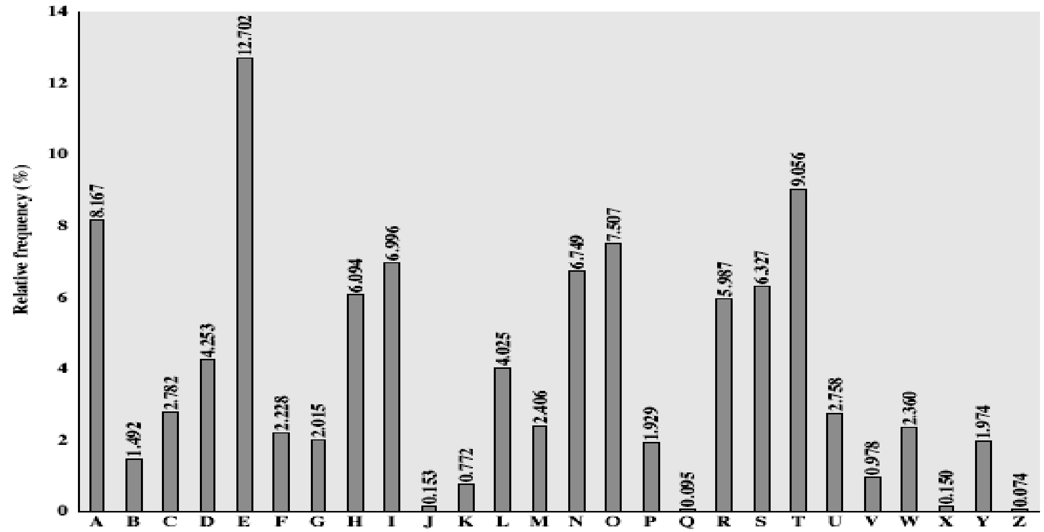
- Main problem with monoalphabetic substitutions is text redundancy
  - **non uniform frequency distributions and correlation**
- In case of human languages:
  - **letter frequencies**
    - in English 'e' is by far the most common letter, then T,R,N,I,O,A,S; other letters are fairly rare (e.g. Z,J,K,Q,X)
  - **two letters frequencies (e.g. “th” in english)**
  - **most common words**
  - **etc.**
- Cryptoanalysis:
  - **discovered by Arabian scientists in 9th century**
  - **calculate letter frequencies for ciphertext**
  - **compare counts/plots against known values**
  - **have tables of single, double & triple letter frequencies**

Cryptanalysis Example

- given ciphertext:

  UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
  VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
  EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

- count relative letter frequencies



- guess P & Z are e and t
- guess ZW is th and hence ZWP is the
- proceeding with trial and error finally get:

  it was disclosed yesterday that several informal but
  direct contacts have been made with political
  representatives of the viet cong in moscow

Polyalphabetic Substitution Ciphers

- An approach to improve security is to use multiple cipher alphabets
  - **polyalphabetic substitution ciphers**
  - **uses a set monoalphabetic substitutions**
  - **defines a rule to determine which cipher alphabet (substitution) should be used at each step**
    - normally uses a key to select which substitution is used for each letter of the message
      - repeat from start after end of key is reached

Vigenère Cipher

- **Simplest polyalphabetic substitution cipher is the Vigenère Cipher (Blaise de Vigenère, 1523-1596)**

plaintext $\mathbf{m = M_0M_1M_2 \ldots M_n}$

$$C_i \leftarrow M_i + K_{i \bmod t} \bmod 26$$

ciphertext $\mathbf{c = C_0C_1C_2 \ldots C_n}$

key $\mathbf{K = K_0K_1K_2 \ldots K_{t-1}}$

- Key is multiple letters long
  - i^{th} letter specifies i^{th} alphabet to use
  - use each alphabet in turn
- Repeat from start after *t* letters
- Effectively multiple Caesar ciphers
  - $\mathbf{K = K_0\ K_1\ \ldots\ K_{t-1}}$
- Decryption simply works in reverse

	0	1	2	3	4	5	6	7	8	9						.	.	.								25
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
+ 1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
+ 2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
+ 3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
+ 4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
+ 5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
+ 6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
+ 7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
+ 8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
+ 9	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
+10	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
+11	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
+12	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
+13	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
+14	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
+15	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
+16	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
+17	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
+18	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
+19	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
+20	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
+21	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
+22	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
+23	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
+24	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
+25	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Vigenère Cipher (cont.)

(00)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
(01)	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
(02)	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
(03)	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
(04)	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
(05)	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
(06)	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
(07)	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
(08)	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
(09)	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
(10)	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
(11)	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
(12)	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
(13)	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
(14)	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
(15)	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
(16)	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
(17)	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
(18)	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
(19)	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
(20)	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
(21)	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
(22)	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
(23)	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
(24)	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
(25)	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

- Example:

  key: REBUS (17,4,1,21,18)

plaintext: **codic emolt osicu ro**

key: **REBUS REBUS REBUS RE**

ciphertext: **TSECU VQPFL FWJWM IS**

Cryptanalysis of Vigenère Ciphers

- Polyalphabetic substitution ciphers make cryptanalysis harder
  - **have multiple ciphertext letters for each plaintext letter**
    - hence letter frequencies are obscured
      - more cipher alphabets to guess and flatter frequency distribution
- But not totally lost
  - **start with letter frequencies**
  - **need to determine number of alphabets (key length)**
  - **then can attack each**

One-Time Pad

- One-Time Pad (OTP) cipher
  - **patented by Gilbert Vernam, Bell Telephone Laboratories, in 1919**
  - **first described by Frank Miller (a California banker) in 1882**
- Can be seen as a special case of Vigenère cipher
  - **the key k={K_0,K_1,K_2,..,K_n} is as long as the plaintext *m***
  - **a random key k is used for each message**
- Ciphertext contains no statistical relationship to the plaintext
  - **for any plaintext and any ciphertext there exists a key mapping one to other**
- In case of alphabet {0,1}, it is: c = m XOR k
  - for each bit: $c_i = m_i$ XOR k_i
  - **two possible substitutions:**
    - $\{0,1\}\rightarrow\{0,1\}$, with $k_i=0$
    - $\{0,1\}\rightarrow\{1,0\}$, with $k_i=1$

One-Time Pad (cont.)

- Example in hexadecimal:

```
m= 48656c6c6f2c20746869732069732061 6e206578616d706c65
```

Example in binary:

```
m= 0100100001100101011011000110110001101111001011000010000001110100 ...
k= 0001010110010101001101011101011000101110110110010100100101100001 ...
c= 0101110111110000010110011011101001000001111101010110100100010101 ...
```

One-Time Pad (cont.)

- Claude Shannon (1945) introduced the definition of perfect secrecy and demonstrated that the one-time pad achieves that level of security
  - **the cipher will be unconditionally secure (unbreakable)**
    - no statistical relationship between distinct ciphertexts
    - for any plaintext of equal length to the ciphertext, there is a key that produces that plaintext
- Disadvantages:
  - **can only use the key once**
    - requires a key stream as long as the sum of all messages that has to be encrypted
      - possible problems on distributing and store this long key

Transposition Ciphers

- Transposition is another classical encryption technique
  - **hides the message by rearranging the letter order (blocks of bits)**
  - **without altering the actual letters used**
  - **performs a sort of permutation**
- Can recognize these since have the same frequency distribution as the original text
- Example
  - **permutation**

Example: Row Transposition Ciphers

- Write letters of message out in rows over a specified number of columns
- then reorder the columns according to some key before reading off the rows

```
Key:        4 3 1 2 5 6 7
Plaintext:  a t t a c k p
            o s t p o n e
            d u n t i l t
            w o a m x y z
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Product Ciphers

- Ciphers using substitutions or transpositions may be not sufficiently secure
- Hence consider using several ciphers in succession to make harder:
  - **two substitutions make a more complex substitution**
  - **two transpositions make a more complex transposition**
  - **a substitution followed by a transposition make a new much harder cipher**
- This is bridge from classical to modern ciphers

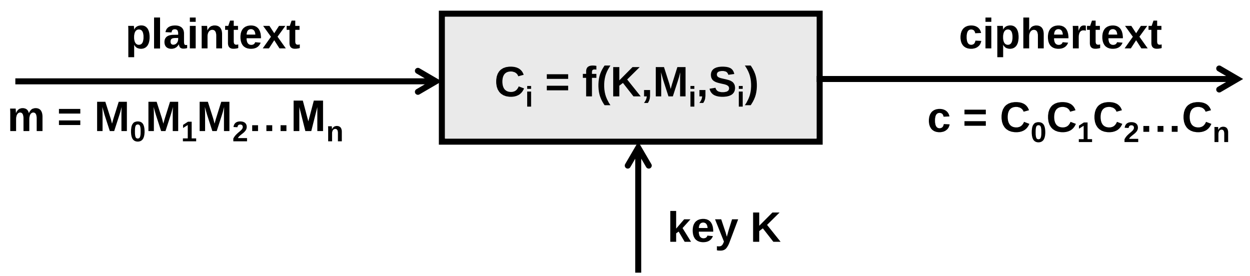
Rotor Machines

- Before modern ciphers, rotor machines were most common product cipher
- Were widely used in WW2
  - **German Enigma, Allied Hagelin, Japanese Purple**
- Enigma uses a series of cylinders, each giving one substitution, which rotated and changed after each letter was encrypted
  - **every key press caused one or more rotors to step by one**
  - **implements a varying substitution cipher**
    - polyalphabetic substitution cipher
- With 3 cylinders have $26x26x26=26^3=17576$ alphabets

Stream and Block Ciphers

Stream ciphers

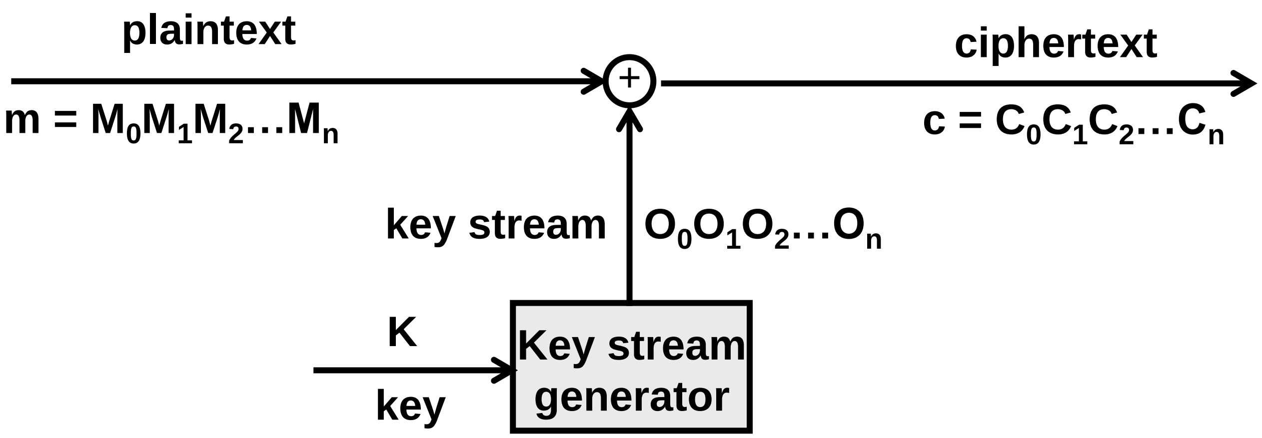
- There are two basic cipher structures
  - **Stream ciphers**
  - **Block ciphers**
- Stream ciphers process messages (Encryption/Decryption) a bit or byte at a time when en/decrypting



- The output unit C_i may be function of the current input M_i, an internal state S_i, and the secret key K
  - **the internal state S_i may be a function of previous M_j and/or C_j, with $j<i$**

Stream ciphers (cont.)

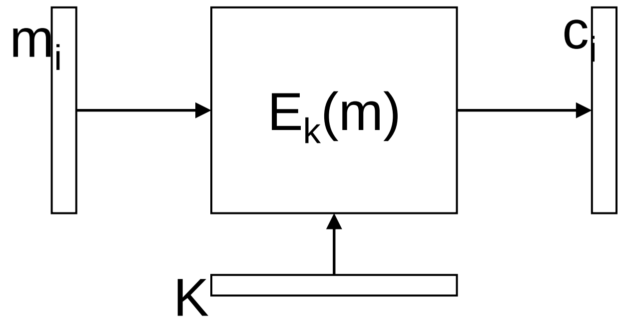
- A very common scheme for stream ciphers (autokey ciphers) is:



- Ideally want a key as long as the message (OTP)
- Most stream ciphers are based on pseudorandom number generators (PRNG)
  - **the key is used to initialize the generator, and either key bytes or plaintext bytes are fed back into the generator to produce the byte stream**

Block ciphers

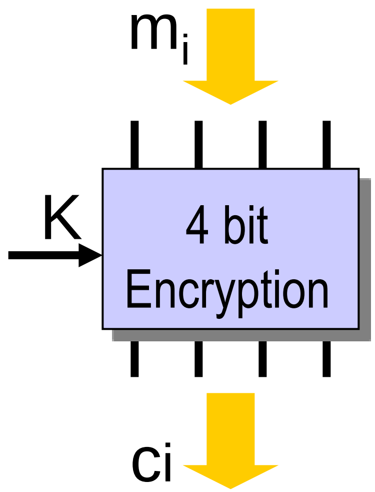
- Block ciphers process messages into blocks, each of which is then en/decrypted
  - **plaintext and ciphertext are treated as a sequence of n-bit blocks of data**
  - **ciphertext is same length as plaintext**
  - **ciphertext depends on plaintext and a key value**



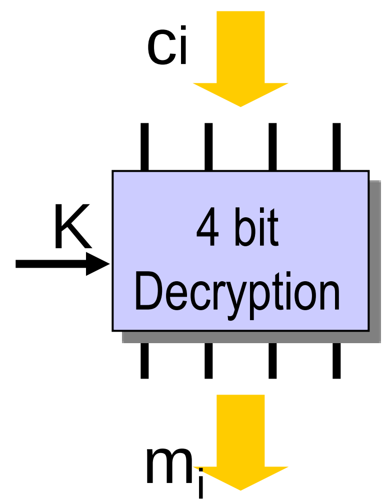
- Larger messages must be processed into blocks, each of which is then en/decrypted
  - **the resulted cipher can be made to behave as a stream cipher**
  - **e.g. CBC, OFB, CFB, and CTR modes**
- Many current ciphers are block ciphers (DES, IDEA, AES, etc.)

Block ciphers (cont.)

- Same input blocks encrypted with the same key are transformed into the same output block
- Example: block size = 4 bits



Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111



Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Block ciphers (cont.)

- Like a substitution on (very big) set of possible inputs
  - **If the block size is *n* bits, 2^n possible input values are mapped to 2^n output values**
    - like a *n*-bit substitution
  - **A way for encrypting could be to specify completely the mapping table (substitution table)**
    - permutation of 2^n *n*-bit inputs
    - there are $2^n!$ different possible transformations
  - **If n=64, would need table of 2^{64} entries storing 64-bit blocks**
    - 2^{64} 2^6 bit = 2^{70} bit = 2^{67}B = 2^{37}TB ≈ 10^{11}TB
    - it is too long
- Instead, a block cipher is created from smaller building blocks and a secret key
  - **if *n* is the cipher size and *k* is the key length, there are a total of 2^k possible transformations, rather than $2^n!$**

Block cipher (cont.)

- How long should the plaintext block be?
  - **having block size too small**
    - in case of known-plaintext attack, an opponent may try to collect $\{M_i,C_i\}$ pairs and construct a decryption table
      - n-bit block cipher requires 2^n pairs
      - It is not required to find the key
    - In case of ciphertext–only attack, if a sequence of M_i have some properties (recognizable sequences of plaintext), it is possible to cryptanalyze the sequence of C_i
      - e.g. exploitation of language redundancy
  - **having block size too long, it could be inconvenient due to the increasing of complexity**
- 64-bit or 128-bit blocks are often used
  - **it is difficult to obtain all 2^{64} pairs (known-plaintext attack)**

Product ciphers

- Product cipher is a type of block cipher that works by executing in sequence a number of simple transformations such as substitution, permutation, and modular arithmetic
- Usually consist of iterations of several rounds of the same algorithm
  - **while the individual operations are not themselves secure, it is hoped that a sufficiently long chain would generate sufficient confusion and diffusion as to make it resistant to cryptanalysis**
- The operation must be reversible
- Important sub-classes of of product ciphers are:
  - **Feistel ciphers**
  - **SP-networks**

Design principles of product ciphers

- Product ciphers are defined in terms of:
  - **Block size**
    - increasing size improves security, but slows cipher
  - **Key size**
    - increasing size improves security, makes exhaustive key searching harder, but may slow cipher
  - **Number of rounds**
    - increasing number improves security, but slows cipher
  - **Subkey generation**
    - greater complexity can make analysis harder, but slows cipher
  - **Round function**
    - greater complexity can make analysis harder, but slows cipher
  - **Performances**
    - fast software en/decryption & ease of analysis

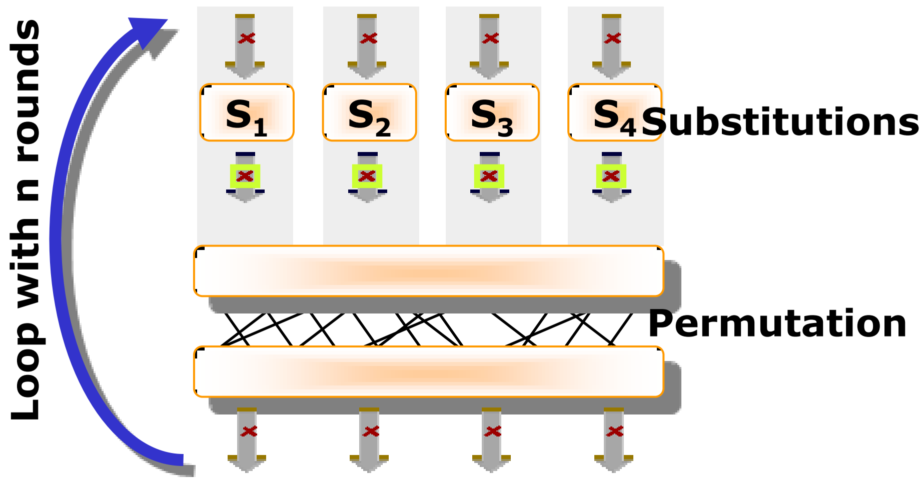
Avalanche Effect

- Key desirable property of encryption algorithms
  - **where a change of one input or key bit results in changing approx half output bits**
    - Avalanche effect
- This is a characteristic of block ciphers
  - **Stream cipher can just have a forward (avalanche) effect**
    - if XOR is used, just the corresponding ciphertext bit is affected
- Modern block ciphers exhibit strong avalanche effect

Advanced Encryption Standard (AES)

Substitution–permutation network

- SP-network is a product cipher that uses only substitutions and permutations
- One possible way to build a block cipher based on SP-network is
  - **break the input into managed-sized chunks (say 8 bits),**
  - **do a substitution on each small chunk,**
  - **and then take the output of all the substitutions and run them through a permuter (big as the input)**
  - **the process is repeated, so that each bit winds up as input to each substitution**
  - **each time is called *round***



Advanced Encryption Standard (AES)

- Block cipher designed to replace DES
  - **organized by National Institute of Standards and Technology (NIST)**
  - **NIST standard on November 26, 2001**
    - FIPS PUB 197 (FIPS 197)
  - **chosen from five candidate algorithms**
    - reviewed by US government (NSA), industry and academia
    - required a four-year process to pick the algorithm
    - winning algorithm chosen Oct 2, 2000
  - **also known as Rijndael block cipher**
    - original name of the algorithm submitted to AES selection process
    - developed by Joan Daemen and Vincent Rijmen (Belgium)
  - **currently, one of the most popular algorithms used in symmetric key cryptography**
    - also adopted as an encryption standard by the U.S. government

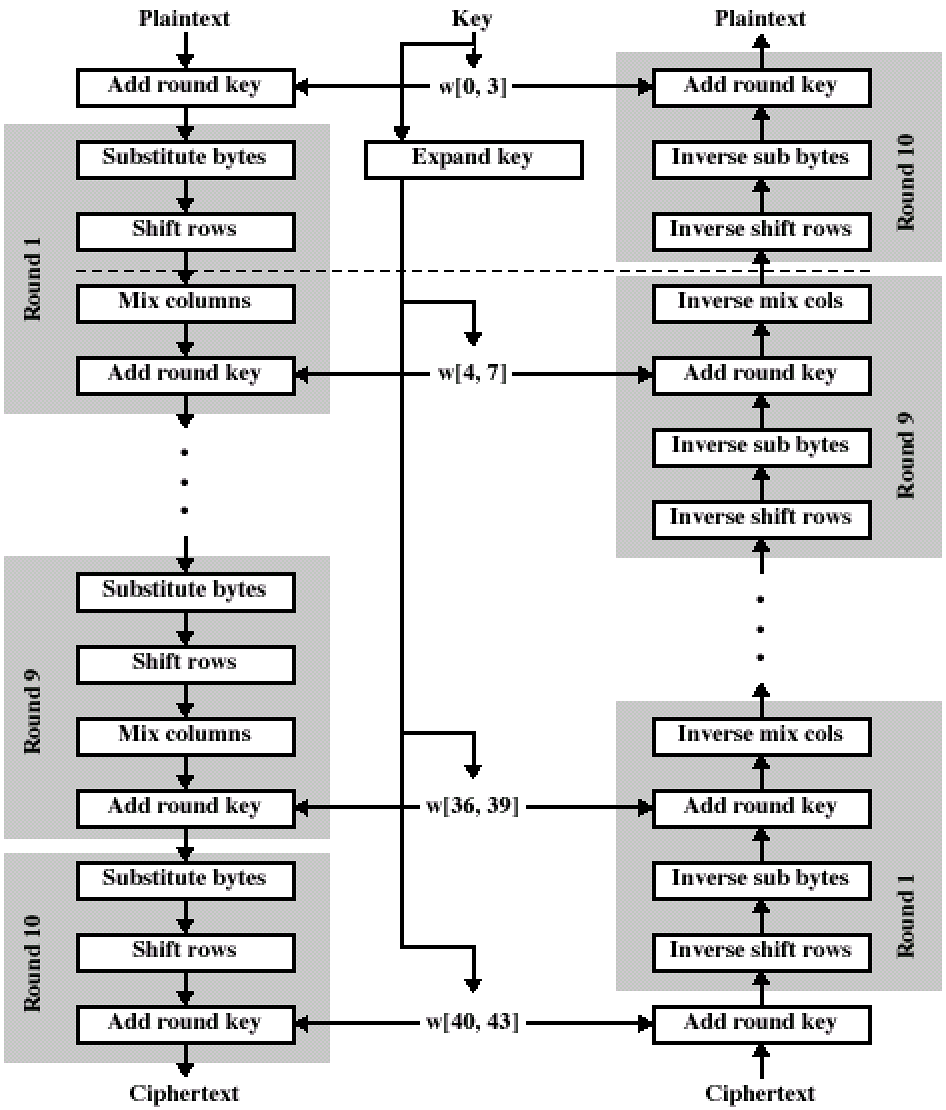
Advanced Encryption Standard (AES) (cont.)

- AES is not precisely the original Rijndael
  - **the original Rijndael algorithm supports a larger range of block and key sizes**
    - Rijndael can be specified with key and block sizes in any multiple of 32 bits, with a minimum of 128 bits and a maximum of 256 bits
- AES has fixed block size of 128 bits and a key size of 128, 192, or 256 bits
  - **i.e. block size: 16 bytes, key size: 16, 24, or 32 bytes**
- Unlike DES, Rijndael is a substitution-permutation network, not a Feistel network
- Fast in both software and hardware
  - **relatively easy to implement**
  - **requires little memory**

AES Description

- Due to the fixed block size of 128 bits, AES operates on a 4×4 array of bytes, termed the state
  - **versions of Rijndael with a larger block size have additional columns in the state**
- AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys
- Most AES calculations are done in a special finite field

AES cipher

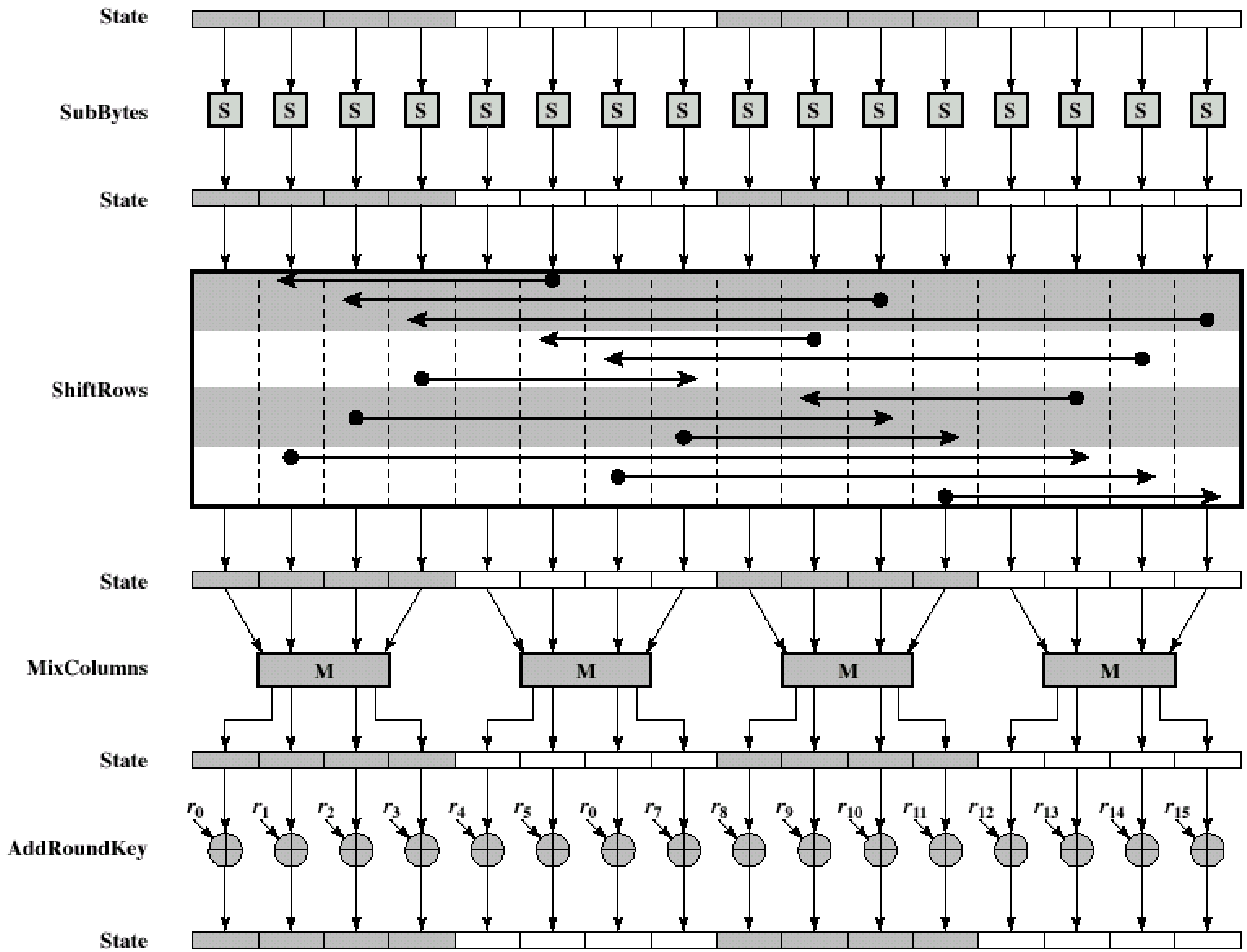


(a) Encryption (b) Decryption

AES cipher

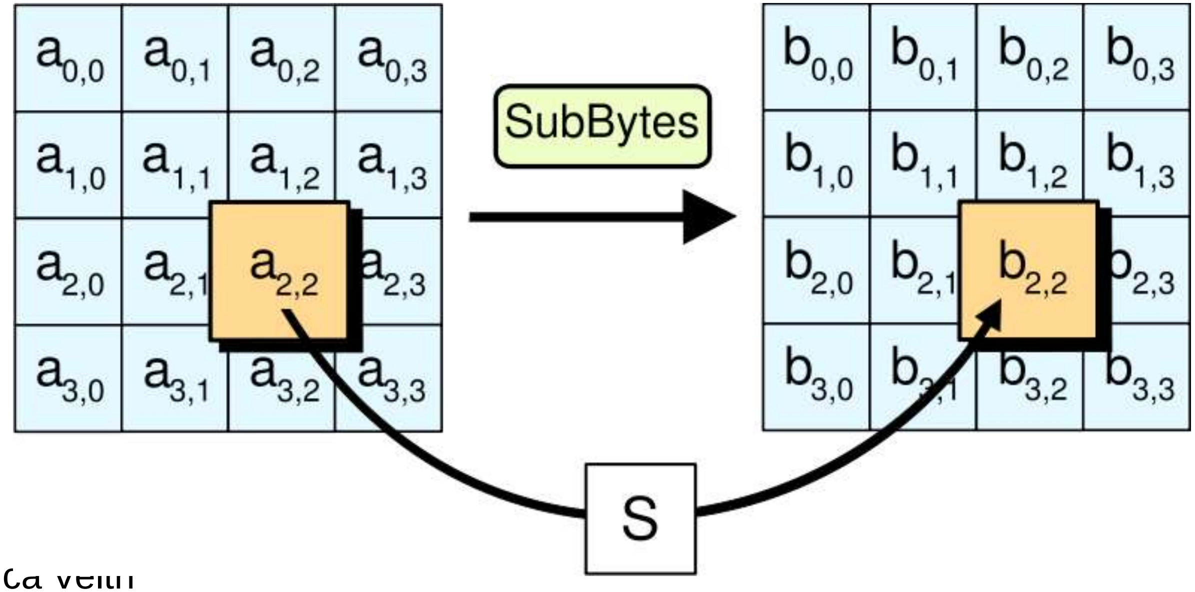
- Algorithm
  - **KeyExpansion using Rijndael's key schedule**
  - **Initial Round**
    - AddRoundKey
  - **(*N*-1) Rounds (9 for 128bit key, 11 for 224bit key, 13 for 256bit key)**
    1. SubBytes — a non-linear substitution step where each byte is replaced with another according to a lookup table
    2. ShiftRows — a transposition step where each row of the state is shifted cyclically a certain number of steps
    3. MixColumns — a mixing operation which operates on the columns of the state, combining the four bytes in each column
    4. AddRoundKey — each byte of the state is combined with the round key derived from the cipher key using a key schedule
  - **Final Round (no MixColumns)**
    1. SubBytes
    2. ShiftRows
    3. AddRoundKey

AES single round



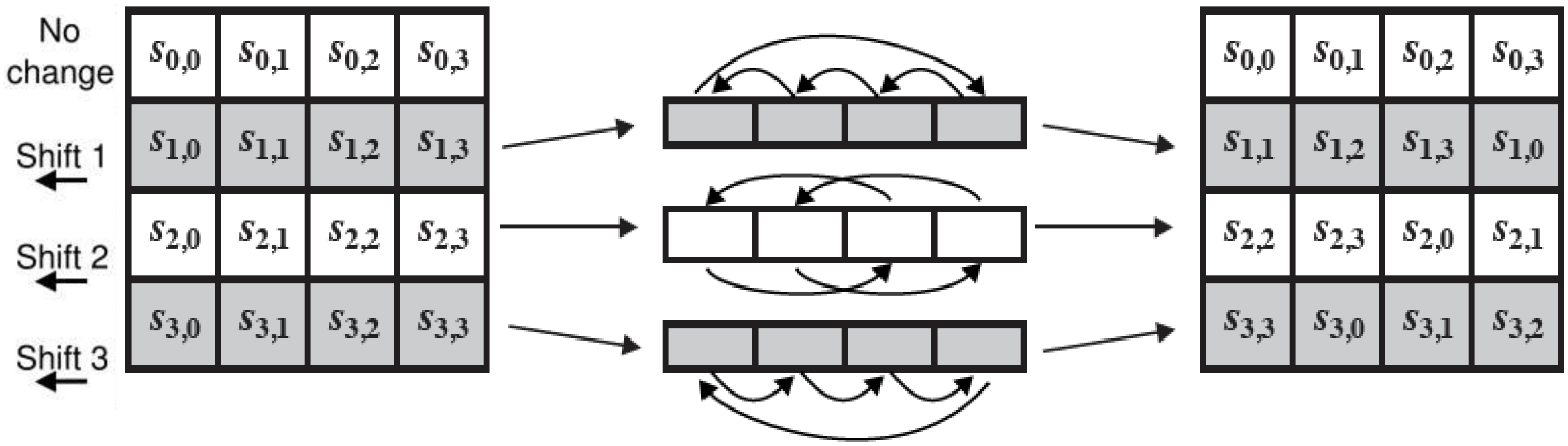
AES SubBytes step

- Each byte in the state is replaced using an 8-bit substitution box, the Rijndael S-box ($b_{ij} = S(a_{ij})$)
  - **S-box can be represented by a table of 256 8-bit values**
    - the trasformation is a simple table lookup
  - **the S-box is derived from the multiplicative inverse over GF(2^8), known to have good non-linearity properties**
    - this operation provides the non-linearity in the cipher



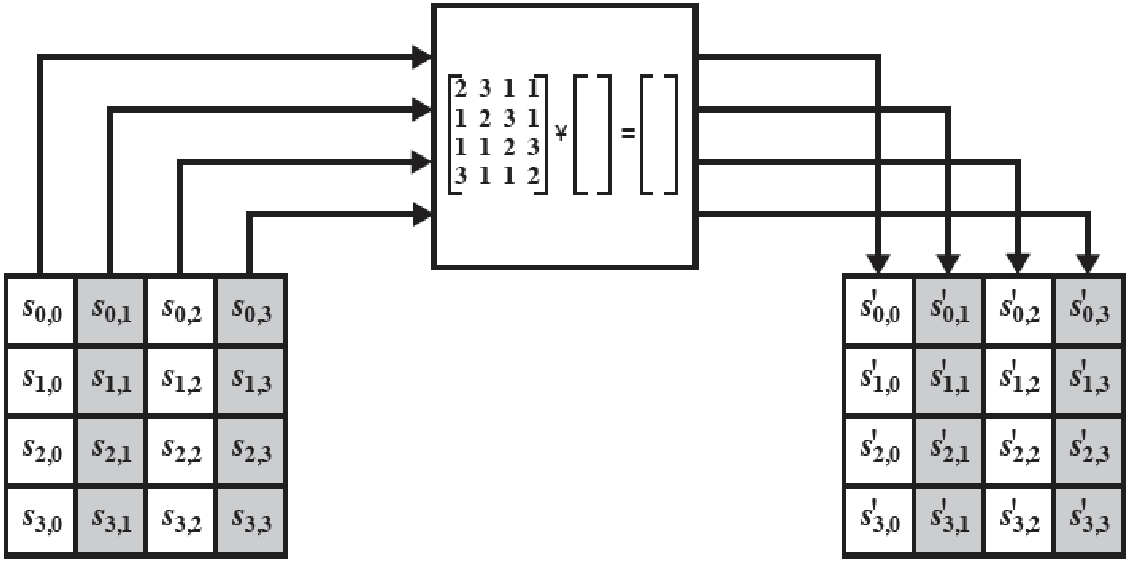
AES ShiftRows step

- ShiftRows step operates on the rows of the state
  - **it cyclically shifts the bytes in each row by a certain offset**
  - **for AES, the first row is left unchanged**
  - **each byte of the second row is shifted one to the left**
  - **similarly, the third and fourth rows are shifted by offsets of two and three respectively**



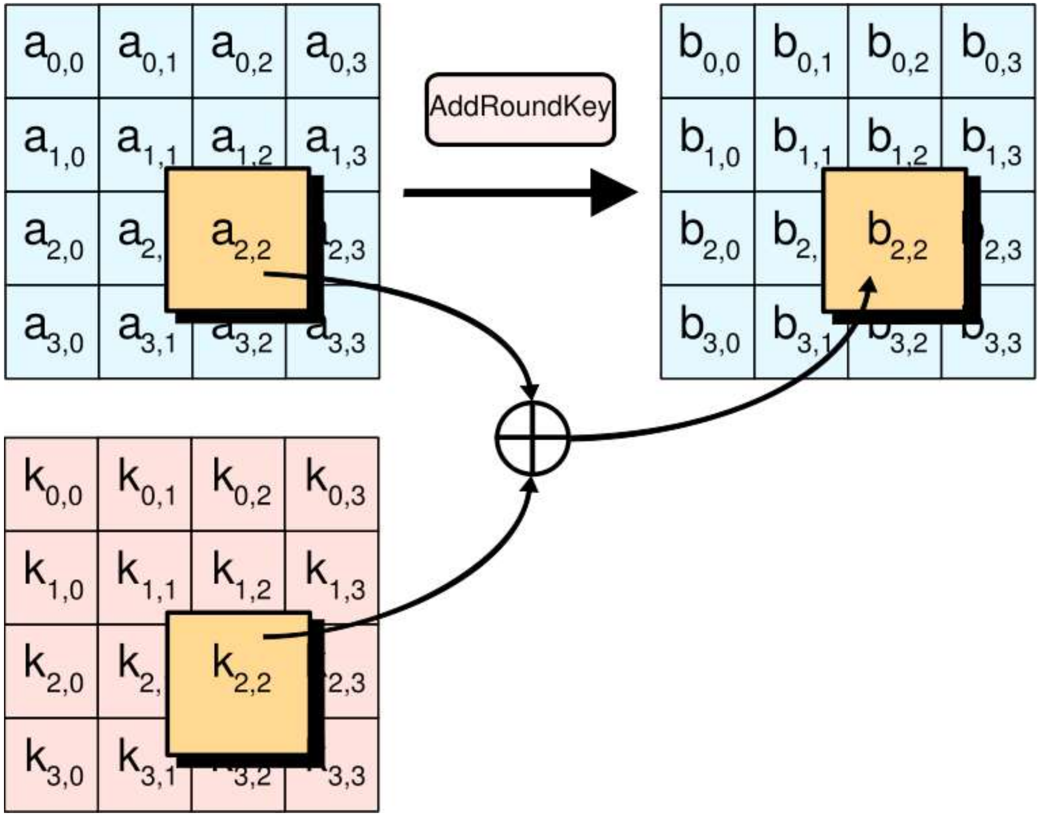
AES MixColumns step

- Four bytes of each column of the state are combined using an invertible linear transformation
  - **each byte of column is a function of all four bytes in that column**
  - **can be viewed as a multiplication *S'=C* x *S* of a particular matrix** C **by the state *S***
    - each column *i* (*i=0,1,2,3*) is treated as a polynomial over GF(2^8) and is then multiplied modulo $x^8+x^4+x^3+x+1$ with an polynomial $c_i(x)$
  - **together with ShiftRows, it provides diffusion in the cipher**



AES AddRoundKey step

- The subkey is combined (XORed) with the state
  - **for each round, a subkey with same size as the state is derived from the main key (key schedule algorithm)**
  - **the subkey is added to the state using bitwise XOR**



AES Security

- In June 2003, the US Government announced that AES may be used also for classified information
  - **this marks the first time that the public has had access to a cipher approved by NSA for encryption of TOP SECRET information**
- The first key-recovery attacks on full AES were published in 2011
  - **the attack is faster than brute force by a factor of about four**
    - it requires $2^{126.1}$ operations to recover an AES-128 key
    - it requires $2^{189.7}$ operations to recover an AES-192 key
    - it requires $2^{254.4}$ operations to recover an AES-256 key
- The only successful attacks against AES have been side channel attacks
  - **however they require a lot of physical information from the system that executes the algorithm**

AES Security (cont.)

- Brute force attack
  - **AES key sizes**
    - 128-bit key
      - $2^{128} = 3.4 \times 10^{38}$ possible keys
    - 192-bit key
      - $2^{192} = 6.2 \times 10^{57}$ possible keys
    - 256-bit key
      - $2^{256} = 1.1 \times 10^{77}$ possible keys
- Comparing to DES, If you could crack a DES key in one second (i.e., try 2^{56} keys per second), it would take 149 trillion years to crack a 128-bit AES key at the same speed
  - **the universe is believed to be less than 20 billion years old**
  - **but, things change**

Building a stream cipher using a block cipher (Block cipher modes)

Encrypting large messages

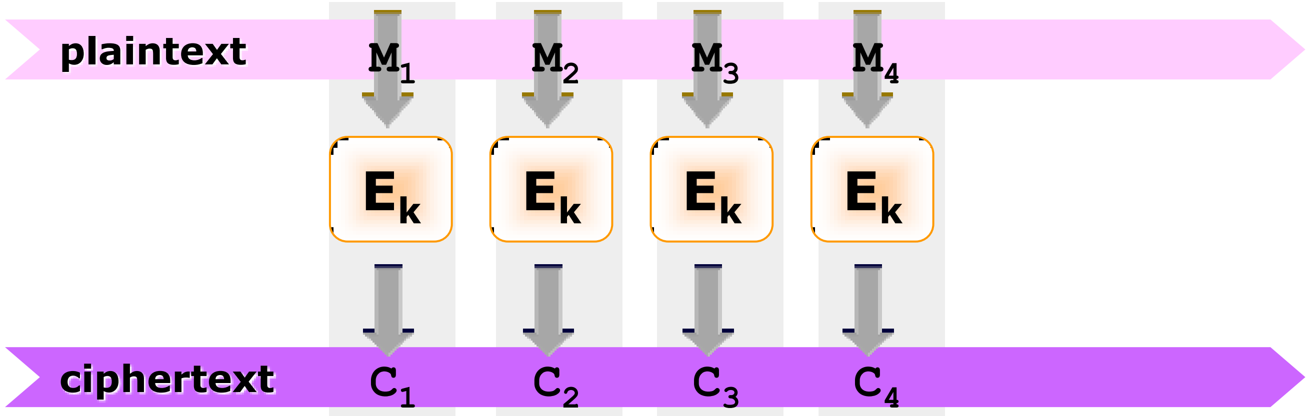
- Block ciphers encrypt fixed size blocks
  - **eg. AES has fixed block size of 128 bits and a key size of 128, 192, or 256 bits**
- Usually we have arbitrary amount of information to encrypt (longer than the block size)
  - **need way to use in practice**
- Five modes have been defined for TDEA in: *NIST Special Publication 800-38A* (2001)
  - **Electronic Code Book (ECB)**
  - **Cipher Block Chaining (CBC)**
  - **(k-bit) Output Feedback (OFB)**
  - **(k-bit) Cipher Feedback (CFB)**
  - **Counter Mode (CTR)**
- These schemes are applicable to any block cipher
- Other modes are also possible

Padding

- If the encryption mode requires that the length of the whole message has to be a multiple of a given block size, a Padding operation has to be performed first
- Examples of padding algorithms:
  - **Bit padding (e.g. ISO/IEC 7816-4)**
    - symbol '1' (bit) is added, and then as many '0' bits as required are added
  - **ANSI X.923**
    - the last byte defines the number of padding bytes; the remaining bytes are filled with zeros
      - eg. [b1 b2 b3 00 00 00 00 05] (3 data bytes, then 5 bytes pad+count)
  - **RFC 5652 Cryptographic Message Syntax / PKCS#7 / PKCS#5**
    - the value of each added byte is the number of bytes that are added
      - eg. [b1 b2 b3 05 05 05 05 05]

Electronic Codebook (ECB) Mode

- Consist of doing the obvious thing, and it is usually the worst method
- The message is broken into n blocks of *b* with padding for the last one
  - **b = block size of the cipher**
  - **n*b = message size including padding**
- Each block is independently encrypted with the secret key K
  - **$C_i = E_K (M_i)$**
  - **$M_i = D_K (C_i)$**



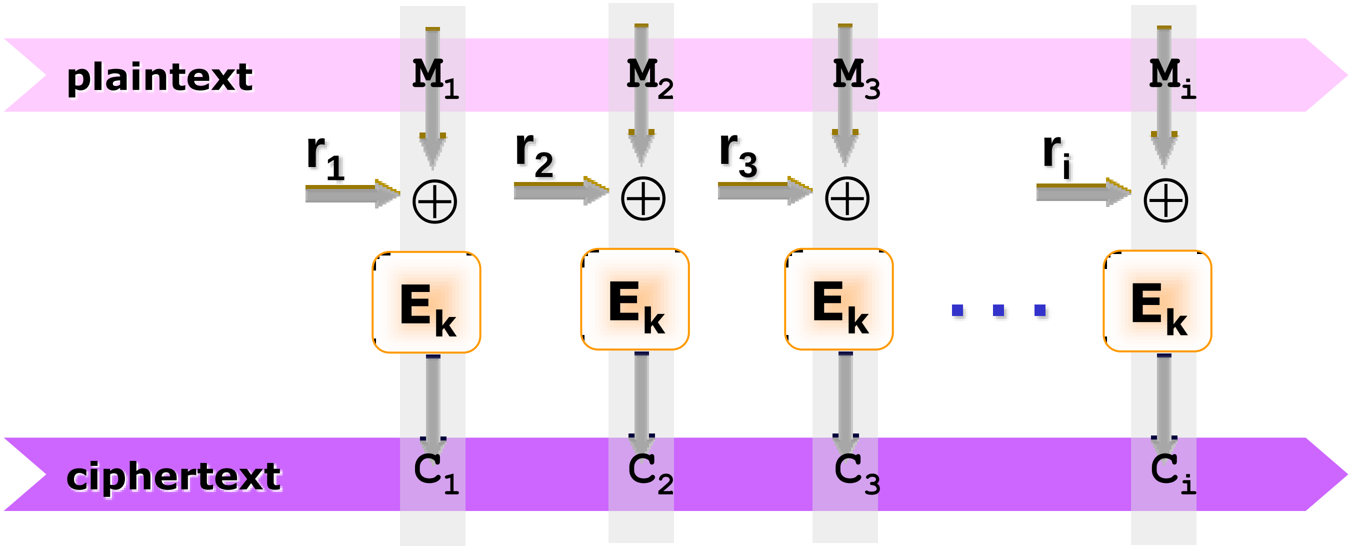
- Each block is a value which is substituted, like a codebook
- The cipher text has the same size of the plaintext (excluding padding)

Advantages and Limitations of ECB

- ECB is very simple and does not introduce extra operation (except for padding)
- There are a number of problems that arise and that don't show up in the single block case
  - **repetitions in message may show in ciphertext if aligned with message block**
    - if a message contains 2 identical blocks, the corresponding cipher blocks are identical; it can be a problem
      - in some cases it can be possible to guess a portion of the message
  - **by comparing two ciphertexts it is possible to discover similarities in the plaintexts**
    - no avalanche effect
  - **(partially) knowing the plaintext, is possible to rearrange the ciphertext blocks in order to obtain a new (known) plaintext**
- As result, ECB is rarely used
  - **main use is sending a few blocks of data (e.g. a secret key)**

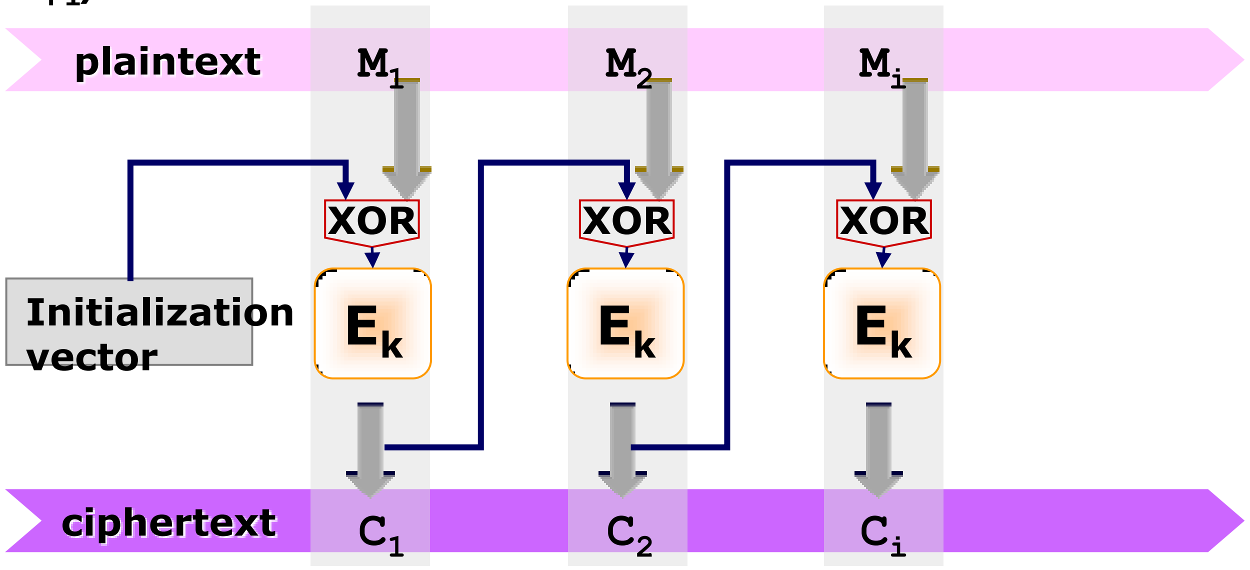
Cipher Block Chaining (CBC) Mode

- CBC objective: if the same block repeats in the plain text, it will not cause repeats of ciphertext
  - **avoiding some problems in ECB**
- How: adds a feedback mechanism to the cipher
- Result: plaintext is more difficult to manipulate
- Basic idea:



Cipher Block Chaining (CBC) Mode

- Plaintext patterns are concealed by XORing a block of *m* with the previous block of *c*
- Requires an IV (Initialization vector) of random data to encrypt the first block
  - $C_i = E_k(M_i \text{ XOR } C_{i-1})$
  - $C_0 = IV$



- Decryption is simple because ⊕ is reversible (A=B⊕C ⇔ A⊕C=B):
  - $M_i = D_k(C_i) \text{ XOR } C_{i-1}$

Advantages and Limitations of CBC

- Each ciphertext block depends on **all** previous message blocks
  - **thus a change in the message affects all ciphertext blocks after the change as well as the original block**
    - forward avalanche effect
  - **a change in IV changes all bits of the ciphertext**
- CBC has the same performance of ECB, except for the cost of generating and transmitting the IV (and the cost of one ⊕)
- It can be used a constant value as IV (e.g. 0), however it can lead to some problems
  - **e.g. if a message is transmitted periodically, it is possible to guess if changes occurred**
  - **if the value is known, attackers may supply chosen plaintext**

CBC Threat 1 - Modifing ciphertext blocks

- Using CBC does not eliminate the problem of someone modifying the message in transit
- If IV is sent in the clear, an attacker can change bits of the first plaintext block (working on the ciphertext and IV) by simply changing the IV
  - **changing bit *h* of IV has predictable effect to bit *h* of M_1**
  - **hence either IV must be a fixed value or it must be sent encrypted in ECB mode before rest of message**
    - e.g. by changing IV $\rightarrow$ IV' = IV$\oplus$X, when decrypting we have:
      $M_1' = D_K(C_1) \oplus IV' = (M_1 \oplus IV) \oplus IV' = M_1 \oplus IV \oplus IV \oplus X = M_1 \oplus X$
- If the attacker changes the ciphertext block C_i, C_i gets $\oplus$'d with the decrypted C_{i+1} to yield M_{i+1}
  - **since M_{i+1} is obtained as: $M_{i+1} = D_K(C_{i+1}) \oplus C_i$**
  - **changing bit *h* of C_i has predictable effect to bit *h* of decrypted M_{i+1}**
  - **however the attacker cannot know the new decrypted M_i' (a new random block, as side effect)**
    - e.g. by changing $C_i \rightarrow C_i' = C_i \oplus X$, when decrypting we have:
      $M_i' = D_K(C_i') \oplus C_{i-1} = D_K(C_i \oplus X) \oplus C_{i-1}$ = unpredictable without knowing *K*
      $M_{i+1}' = D_K(C_{i+1}) \oplus C_i' = (M_{i+1} \oplus C_i) \oplus C_i' = M_i \oplus X$

CBC Threat 2 - Rearranging ciphertext blocks

- Knowing the plain text, the corresponding ciphertext and IV, it is possible to rearrange the C_1, C_2, C_3, .. (building blocks), in such a way to obtain a new known M'_1, M'_2, M'_3 ..
  - **e.g. suppose an intruder rearranges the plaintext $c=C_1,C_2,C_3,C_4$:**
    - if the modified ciphertext $c'=C_1,C_3,C_3,C_4$, then when decrypting:
      $M_1' = D_K(C_1') \oplus IV' = D_K(C_1) \oplus IV = M_1$
      $M_2' = D_K(C_2') \oplus C_1' = D_K(C_3) \oplus C_1 = (M_3 \oplus C_2) \oplus C_1$
      $M_3' = D_K(C_3') \oplus C_2' = D_K(C_3) \oplus C_3 = (M_3 \oplus C_2) \oplus C_3$
      $M_4' = D_K(C_4') \oplus C_3' = D_K(C_4) \oplus C_3 = M_4$
    - if the modified ciphertext $c'=C_1,C_3,C_2,C_4$, then when decrypting:
      $M_1' = D_K(C_1') \oplus IV' = D_K(C_1) \oplus IV = M_1$
      $M_2' = D_K(C_2') \oplus C_1' = D_K(C_3) \oplus C_1 = (M_3 \oplus C_2) \oplus C_1$
      $M_3' = D_K(C_3') \oplus C_2' = D_K(C_2) \oplus C_3 = (M_2 \oplus C_1) \oplus C_3$
      $M_4' = D_K(C_4') \oplus C_3' = D_K(C_4) \oplus C_2 = (M_4 \oplus C_3) \oplus C_2$
- These threads can be combated by adding a MIC (message integrity check) code, or a strong checksum to the plaintext before encrypt
  - **use of 32 bit checksum doesn't completely solve the problem, since 1 in 2^{32} chance that the checksum will work**
    - after 2^{31} attempts, the probably of obtaining a correct checksum is ~50%

Output Feedback (OFB) Mode

- Acts like a pseudorandom number generator
  - **more precisely as a stream cipher based on XOR**
- The message is encrypted by ⊕ing it with the pseudorandom stream generated by the OFB
  - **message is treated as a stream of bits**
- How it works:
  - **A pseudorandom number O_0 is generated (named IV as in CBC)**
  - **O_0 is encrypted (using secret key K) obtaining O_1**
  - **from O_1 is obtained O_2 and so on, as many block are needed**

    $O_i = E_K(O_{i-1})$

    $O_0 = IV$
  - **the pseudorandom stream (like a one-time pad) is independent of message and can be computed in advance**
  - **the one-time pad is simply ⊕'d with the message**

    $C_i = M_i \text{ XOR } O_i$
- Example of uses: stream encryption over noisy channels

Output Feedback (OFB) Mode

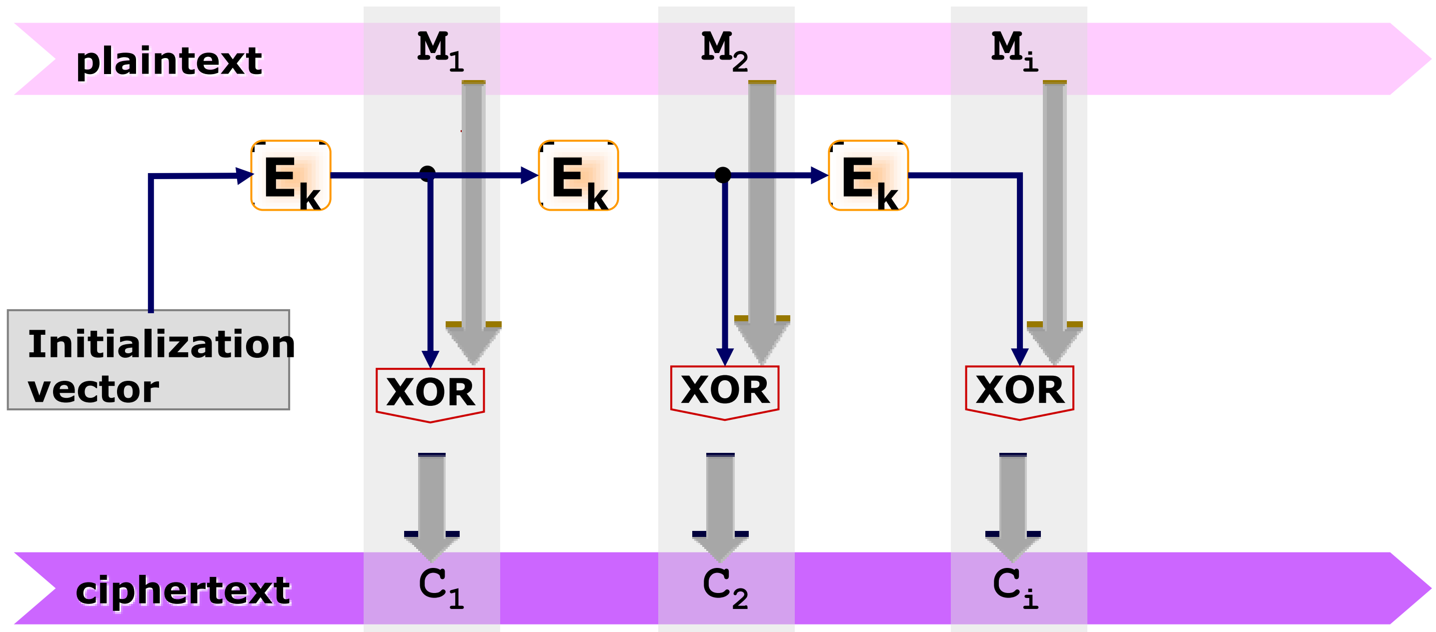
- OFB in short:
  - **a long pseudorandom string is generated (one-time pad)**

    $O_i = E_K(O_{i-1})$

    $O_0 = IV$
  - **the one-time pad is ⊕'d with the message**

    $C_i = M_i \text{ XOR } O_i$

    $M_i = C_i \text{ XOR } O_i$



Advantages and Limitations of OFB

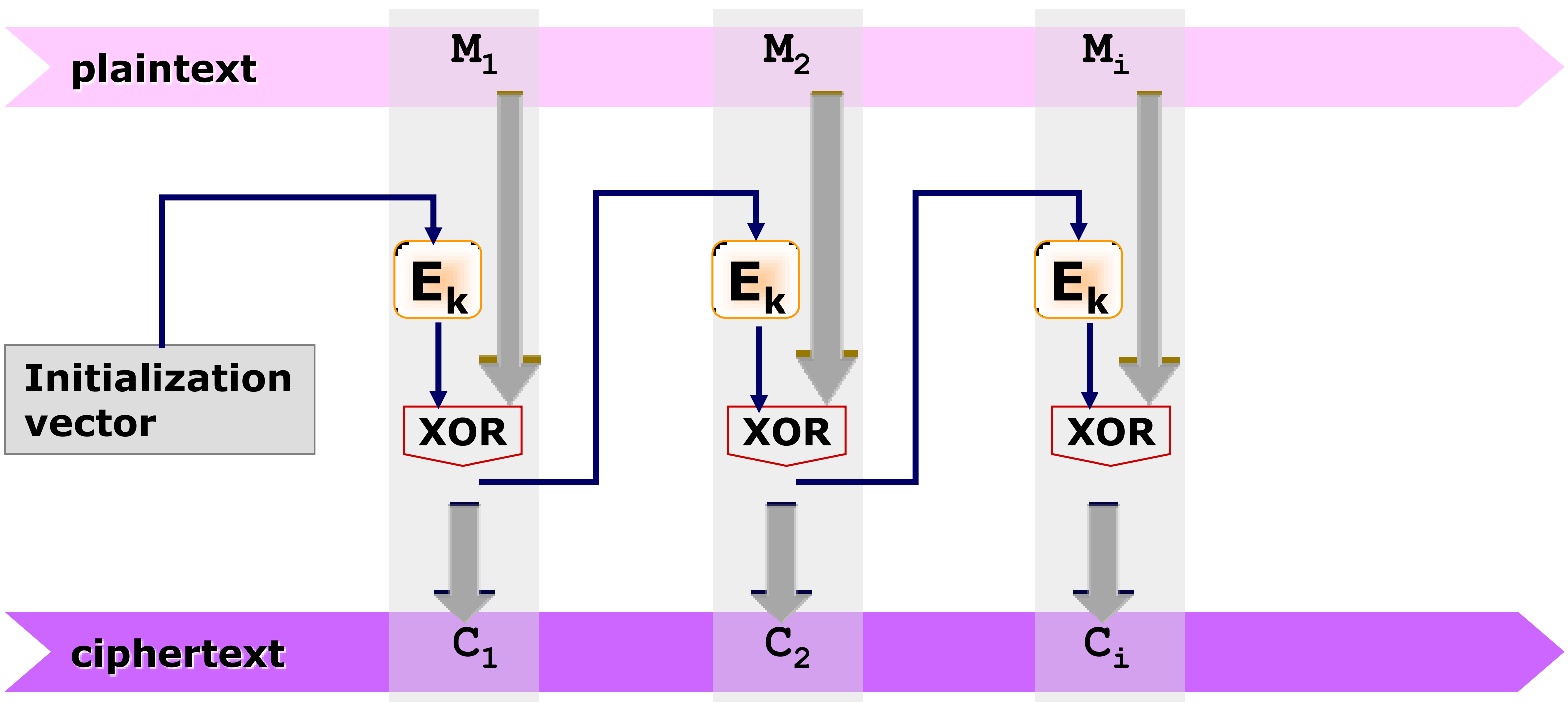
- Advantages of OFB:
  - **one-time pad can be generated in advances**
  - **if a message arrives in arbitrary-sized chucks, the associated ciphertext can immediately be transmitted**
  - **possibility to encrypt variable-length messages: no need of padding**
  - **if some bits of the ciphertext get garbled, only those bits of plaintext get garbled (no error propagation)**
- Disadvantages of OFB:
  - **if the plaintext is known by a bad guy, he can modify the ciphertext forcing the plaintext into anything he wants**
  - **hence must never reuse the same sequence (key+IV)**
  - **sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs**

Cipher Feedback (CFB) Mode

- Similar to OFB
- The *b* bits shifted in to the encryption module are the *b* bits of the ciphertext from the previous block

  $C_0 = IV$

  $C_i = M_i \text{ XOR } E_K(C_{i-1})$



Advantages and Limitations of CFB

- The block cipher is used in encryption mode at both ends
  - **like in OFB**
  - **however:**
    - the one-time pad cannot be generated entirely in advance
      - needs to stall while do block encryption after every *b* bits
- Errors in cipher text propagate to the next block
  - **like in CBC**
- The lost of a portion of the ciphertext can be resumed if it is multiple of s-bit used by the CFB
  - **it is possible to have *s*-bit CFB with *s* different from *b* (i.e. the $E_k(\cdot)$ size), e.g. 8 bits**
    - with OFB or CBC if octects (bytes) are lost in transmission or extra octects are added, the rest of transmission is garbled
    - with 8-bit CFB as long as an error is an integral number of octects, things will be resynchronized

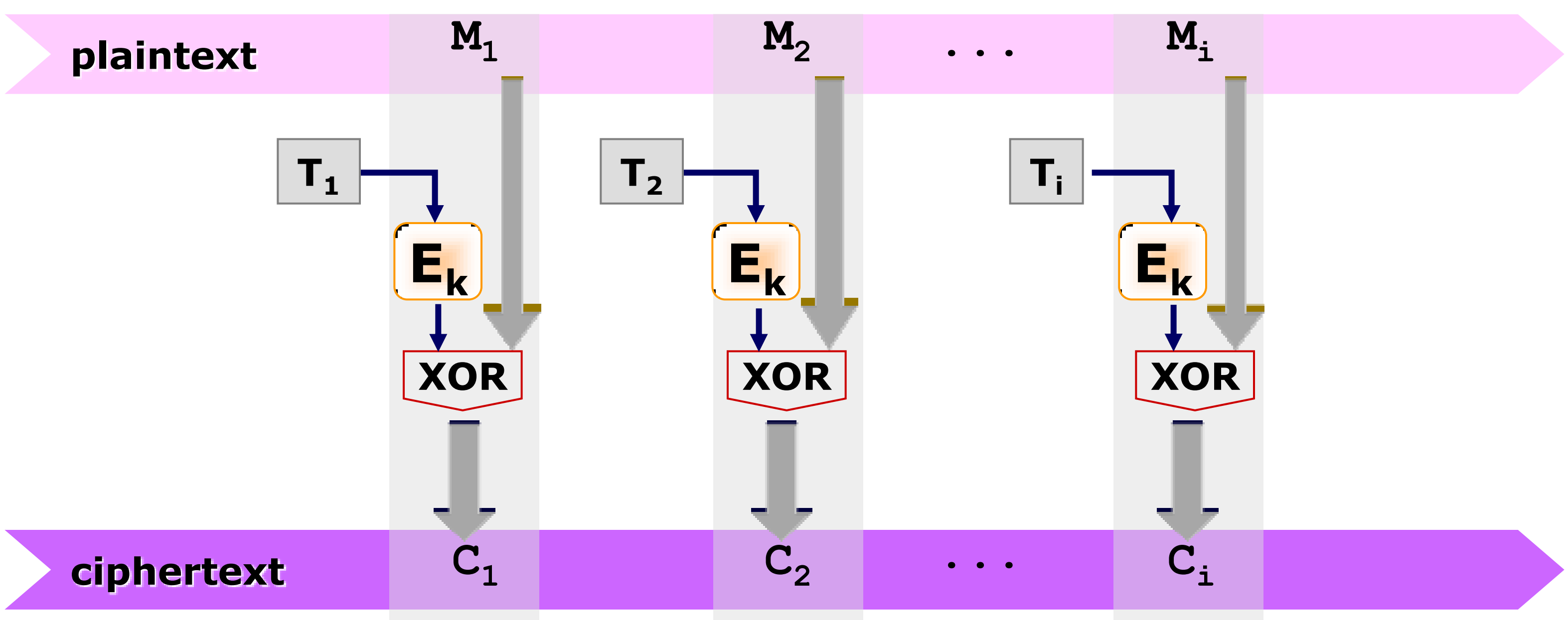
Counter (CTR) Mode

- Similar to the OFB
- The CTR output blocks are generated by encrypting a set of input blocks T_i called counters

  $O_i = E_K(T_i)$

  $C_i = M_i \text{ XOR } O_i$

  - **the sequence of counters $T_1, T_2, .., T_i$ must have the property that each block in the sequence is different from every other block**



Counter (CTR) Mode (cont.)

- Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block (modulo 2^n, where n is the block size)
- Advantages:
  - **the forward cipher functions can be applied to the counters prior to the availability of the plaintext or ciphertext data**
    - like OFB
  - **the cipher functions can be performed in parallel**
  - **the plaintext block that corresponds to any particular ciphertext block can be recovered independently from the other plaintext blocks if the corresponding counter block can be determined**